HAL
open science

First steps toward a Verification and Validation Ontology

Mounira Kezadri, Marc Pantel

To cite this version:

Mounira Kezadri, Marc Pantel. First steps toward a Verification and Validation Ontology. Embedded Real Time Software and Systems (ERTS2012), Feb 2012, Toulouse, France. hal-02189894

HAL Id: hal-02189894

https://hal.science/hal-02189894v1

Submitted on 20 Jul 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

First steps toward a Verification and Validation Ontology

Mounira KEZADRI and Marc PANTEL

Université de Toulouse, IRIT - France
First name.Last name@enseeiht.fr

Abstract. This paper presents the key elements of an ontology that formalizes part of the knowledge about behavioural modeling and the associated verification and validation technologies. It summarizes the concepts existing in this area of interest and the relationships among them. We propose a classification of different modeling formalisms and a representation of possible verification and validation methods. A system is represented using several views conforming to different modeling languages. Its properties can be assessed with verification and validation technologies. We also describe existing $V\&V$ tools and how they are related to the other elements.

Keywords: Ontology, Modeling languages, Verification and Validation.

1 Introduction

Verification and validation ($V\&V$) are activities in a development process checking that a system satisfies its requirements. Verification relates a system implementation and its explicit specification whereas validation relates a system and its end users implicit requirements. If the requirements have been specified, then the system is verified against this specification, and the specification is validated against its end users needs.

Given the huge number of $V\&V$ methods and tools, a formal description of the $V\&V$ domain is mandatory in order to ease the choice of the best technology when a developer needs to verify and/or validate parts of the system he/she is building. This paper presents the preliminary elements of a $V\&V$ Ontology (VVO). It represents a knowledge sharing initiative for the $V\&V$ domain and provides a formal representation of the key data regarding this domain.

The VVO contains a knowledge base produced by populating the classes of the ontology with informations and appropriately relating them. The classification and the initial population of the VVO are mainly based on the state of the art of the CESAR[1] and Ptolemy[2] projects. The VVO ontology defines the verification and validation methods that can be applied on a system whose

[1] https://cesarproject.eu/
[2] http://ptolemy.eecs.berkeley.edu/

behavior has been modeled using the modeling languages that are also defined in the *VVO*. It defines the semantic relations between *V*& *V* tools, formalisms and views and serves as a platform to integrate other tool specifications. *VVO* will be used as a communication language, as a foundation for other engineering ontologies, and later to constitute a global verification and validation platform for a set of *V*& *V* techniques. This knowledge base will serve as a basis to guide the choice of the *V*& *V* tool related to the formalism used to model the system. Our current implementation spans three domains namely: views, formalisms, and *V*& *V* techniques.

In this paper we describe the conceptualization of the ontology, and motivate the major representation choices. Our contributions are: 1) classification of the systems' description formalisms, 2) classification of the Verification and Validation methods, 3) definition of the relations between the *V*& *V* technologies, formalisms and properties description languages. Our intention is first to make the knowledge of the behavior modeling and *V*& *V* domains shareable and reusable and then to use the *VVO* as a guideline for choosing and applying dynamically the adapted *V*& *V* technologies in order to ease the development of safety critical systems. The *VVO* contains more than 250 classes. It is available for reuse, comments and extension proposals at `http://www.irit.fr/~Mounira.Kezadri/Ontologies/VVO.owl`.

The remainder of this paper is organized as follows: In Section 2 the ontology development methodology, the ontology language choice, the general architecture of the *VVO*, the definition of global concepts and relations between them are given. In Section 3 we present the case study of verifying Petri Nets. Section 4 discusses related work. Conclusion and future work are given in Section 5.

2 The *VVO* construction

In this section we present first, the development methodology, then the language and development tool choices and last, the general architecture of the ontology. We have chosen the Web Ontology Language (OWL) [McGuinness et al., 2004] in the Protégé toolset[3] (version 4.1 Alpha) to build the *VVO* and the Jambalaya [4] and OWLViz [5] tabs (version 3.4.3) to generate the figures.

2.1 The ontology development methodology

A model of a complex real word domain such as *V*& *V* can be explicitly represented with existing objects, entities and relationships between them. As in the majority of domains, it is a complex and challenging task. As explained in [Fonou-Dombeu and Huisman, 2011], a rigorous construction of an ontology requires the use of the development methodologies and platforms. A number of ontology modeling methods [Gomez-Perez et al., 1991] have been proposed in

[3] http://protege.stanford.edu/
[4] http://protegewiki.stanford.edu/wiki/Jambalaya_2.6.0
[5] http://protegewiki.stanford.edu/wiki/OWLViz

the literature. A methodology mainly prescribes guidelines for the specification, conceptualization, formalization and implementation of the ontology. The specification phase defines the aims and roles of the intended ontology, as well as the people who will be using it. During the conceptualization phase, a conceptual or domain ontology is built. In its simple representation, a conceptual or domain ontology is a graph where the vertices are objects, concepts and entities of the domain, and the edges are lines interconnecting pairs of vertices and representing the relationships between the constituents of the domain, in our case the diagram shown in Figure 1. During the implementation phase of the ontology development, the ontology is formally represented in one of Semantic Web languages with ontology editing platforms.

2.2 The ontology language choice

We have selected a language which is mature, standardized, for which there is tool support for the design and maintenance, and which is expressive enough to design the domain. This language is OWL2. The Web Ontology Language, informally OWL2, is an ontology language for the Semantic Web with a formally defined meaning. OWL2 ontologies provide classes, properties, individuals and data values which are stored as Semantic Web documents. OWL2 is a W3C recommandation [Motik et al., 2009] since 2009.

2.3 The ontology development tool choice

Protégé was selected from the existing ontology development tools. It is an open-source platform that is widely used all over the world and is mature, scalable and extensible. It provides an interactive graphical interface for ontology design, display, and manipulate. Its internal structure represents ontology elements as classes, properties, constraints, and instances. Protégé can be used to load, design, edit and save ontologies in OWL and different other formats.

2.4 The general structure of the *VVO*

The first phase for the ontology definition is enumerating the important terms and relationships of the $V\&V$ domain.

Figure 1 shows the global structure of our ontology, it represents its key concepts and the relations between them. The main concepts are: $Formalism$, $System$, $Properties$, VV, $SystemAbstraction$, $View$ and $ViewPoint$. The principle properties or relations used to link these concepts are: $describes$, $verifies$, $concerns$, $checks$, $specifies$ and $conforms$.

As shown in Figure 1, a system can be described using different abstractions. Every abstraction is expressed using a modeling language conforming to a view. We can associate properties to systems and system abstractions. The system with, or without, associated properties can then be assessed using some $V\&V$ technique.

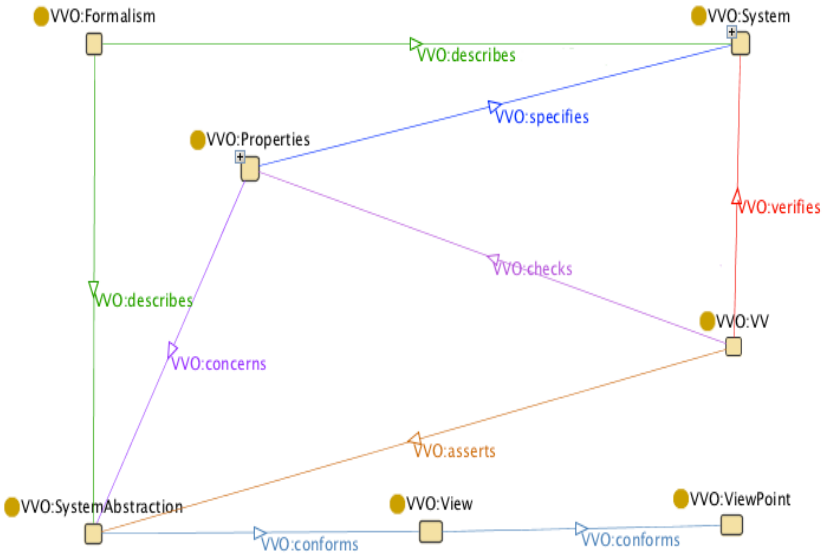


Figure. 1: Global architecture of *VVO*.

In the following subsections, we present the definition and the hierarchy of the main concepts of the *VVO*. It is constituted from three sub-ontologies: an ontology for views, an ontology for formalisms, and an ontology for *V*& *V* techniques.

An ontology for views: Successful modern system development requires to consider multiple views. The challenge is to combine, coordinate and manage the systems' abstractions according to this views.

The aim of this sub-ontology is to describe the views domain. Its concepts are mainly derived from the IEEE 1471 standard [Hilliard, 2000].

System Abstraction concept: Modern systems are generally very complex and several abstractions are usually required in order to manage their description. An abstraction conforms to a view and it is described using some formalism, it can be characterised by some properties that must be verified using some *V*& *V* technique.

ViewPoint concept: Is a specification of the conventions used for constructing and exploiting a view. It is also a pattern or template from which to develop individual views by establishing the purposes, rules, and audience for a view and the techniques for its creation and analysis. Figure 2 shows a part of the Viewpoint classification in the *VVO*, it is non-exhaustive and can be linked with elements from the View hierarchy. The standard is also agnostic about where viewpoints come from. Contributions for enrichment are encouraged.

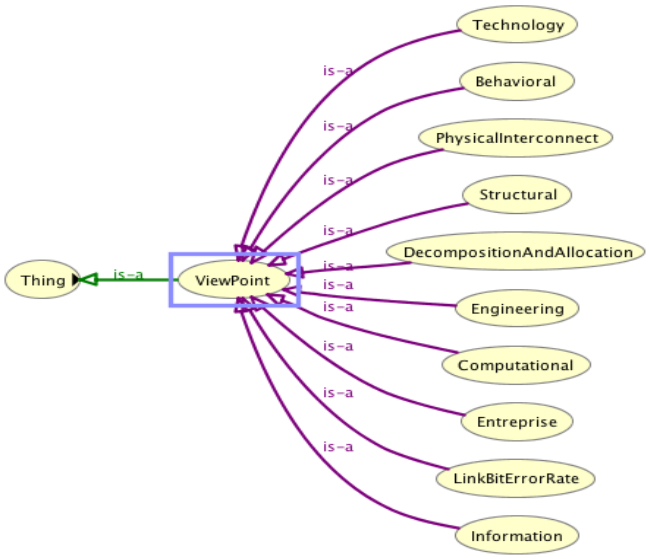


Figure. 2: ViewPoint concept hierarchy.

View concept: The representation of a whole system from the perspective of a related set of concerns is called a View. According to the standard, a View must be conforming to at least one ViewPoint. Hence, the elements from the ViewPoint hierarchy must be linked to elements from View hierarchy. This is represented by the relation $conforms$ in Figure 1. The hierarchy of Views is not detailed in this paper. An interested reader can download the ontology for more information.

An ontology for description formalisms: A formalism can describe systems and system abstractions. We consider this part of the ontology as a basis for structuring and constructing domain-specific modeling tools. The principle concepts linked to this ontology are:

System concept: A system is a collection of components organized to accomplish a specific function or a set of functions. One system can be composed of one or more components. It is described using several views expressed in different modeling formalisms (the $describes$ relation in Figure 1). We can associate some properties with the system (the $specifies$ relation in Figure 1). Finally, the system is verified using some $V\&V$ technique (the $verifies$ relation in 1).

Formalism concept: The Formalism ontology collects a large number of widely-used formalisms for modeling systems' behavior. As the number of formalisms is quite important, we propose a classification of these formalisms, a small part of which is shown in Figure 3.

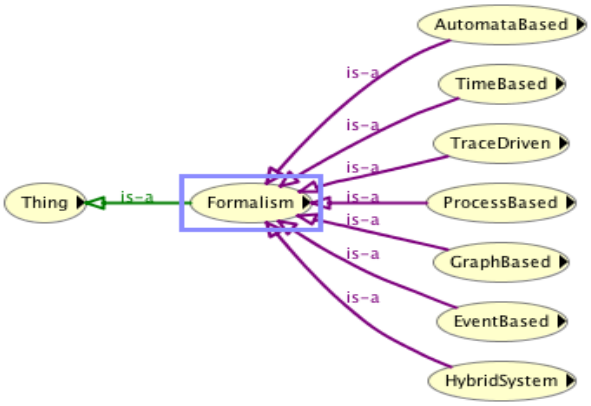


Figure. 3: Part of the first level of formalisms hierarchy.

We present in Figure 4 an example of a not exhaustive automata formalism hierarchy. Automata is a sub-class of automata-based formalisms, it is composed from: hybrid automata [Alur et al., 1993], Büchi automata [Büchi, 1960], hierarchical automata [Mikk et al., 1997], finite automata [Lawson, 2005], Muller automata [Perrin, 2004], cellular automata [Cervelle et al., 2010], timed automata [Bengtsson and Yi, 2003] and stochastic automata [D'Argenio and Katoen, 2005]. Each kind of formalism has its special characteristics and can have its own hierarchy.

An ontology for Verification and Validation techniques: The correctness of a system with respect to the desired behavior is verified using some $V\&V$ method, which checks whether the structure that models the system satisfies a formula describing that behavior. This formula is called *Property*. The most important and largest part of this work is the classification of the $V\&V$ techniques.

V&V *concept:* A wide variety of $V\&V$ strategies and techniques are available. A $V\&V$ technique (method or technology) can be applied to one or several abstractions of a system depending on the formalism used to describe the system and the property that must be assessed. By verification techniques, we mean a technique to verify that a system satisfies some specification in some measure. By validation technique, we mean a technique for the modeling language user to check that the model is a correct rendering of the idea he wanted to express. The goal is to increase the confidence that we have on the developed system. This can be done with different approaches.

Figure 5a shows a very small part of the hierarchy that we propose for the $V\&V$ techniques. This hierarchy is split to several levels. The Analysis concept, for example, has 60 sub-categories.

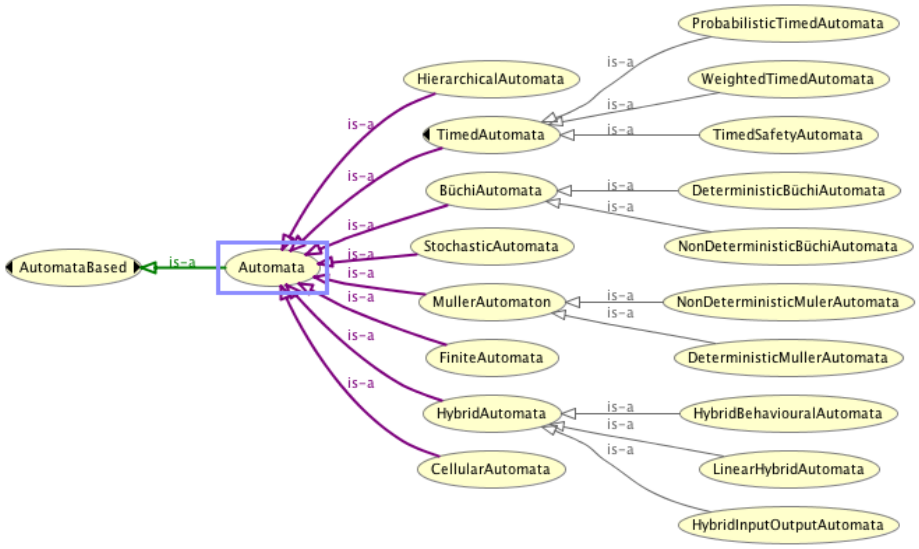
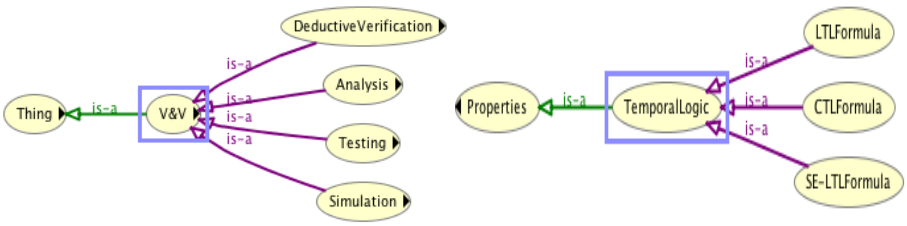


Figure. 4: Automata based hierarchy.

Property concept: To describe a property, we can use several Property Description Languages (PDL). As an example we present the part of temporal logics that we use in the case study in Section 3. We can differentiate several temporal logics, the Linear Temporal Logic (LTL), Computational Tree Logic (CTL) and State Event-LTL (SE-LTL) are subclasses of temporal logic presented in Figure 5b. A SE-LTL formula [Chaki et al., 2004] can be associated to some model described in the Petri Nets formalism (the hierarchy of Petri Nets is illustrated in Figure 5). This kind of property associated to Petri Nets can be verified using a model checking technique.



(a) $V\&V$ techniques Hierarchy. (b) Temporal Logics Hieradrchy.

3 Case study

We instantiated the *V&V* concepts of the ontology with several *V&V* tools, such as the TINA toolbox used in this case study. To elaborate this test, the DL Query[6] tab in Protégé is used. We present the use case for the verification of Petri Nets, also known as place/transition nets. Petri Nets is one of the formalisms classified in *VVO*, it is composed of several sub-classes, a part of the Petri Nets hierarchy is presented in Figure 5.

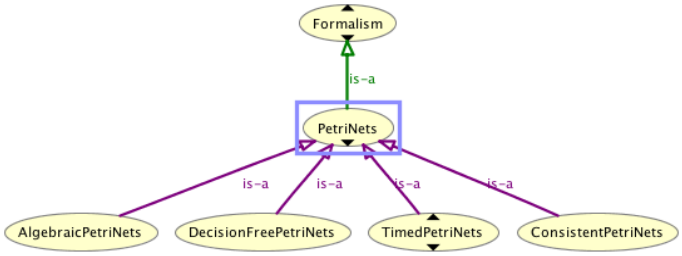


Figure. 5: Petri Nets hierarchy.

Thanks to DL_Query, querying the *VVO* ontology is simple. For example, if we want to verify a SE-LTL (State/Event LTL) formula described on a time Petri nets system, the query and its result are shown in Figure 6.

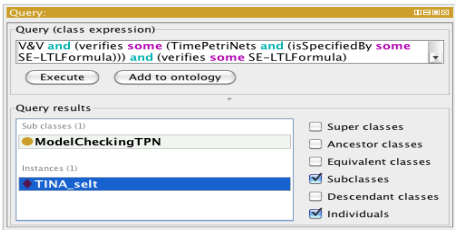


Figure. 6: A query example.

The query states that the *V&V* technique must support the time Petri nets formalism and the SE-LTL formula and that a property on the Time Petri nets formalism can be defined in the SE-LTL logic as presented in Figure 6. The result is TINA-Selt, the *V&V* tool specialized in the verification of SE-LTL formulas on +Time Petri nets systems.

[6] http://protegewiki.stanford.edu/wiki/DL_Query

4 Related work

By domain ontology we mean an ontology that concerns a specific domain of interest. It defines the basic terms and relations comprising the vocabulary of the area, as well as the rules for combining terms and relations to define extensions to the vocabulary. For instance, a number of domain ontologies are available on the Internet, covering several areas including medicine (eg: Ménélas[7]), world government and legal Knowledge Systems [Visser and Bench-Capon, 1998], Biomedical [Bodenreider and Burgun, 2005], Economic and Financial Information Management [Castells et al., 2004], and biological viruses (BVCO). To the best of our knowledge, there is no ontology reported anywhere for the *V*&*V* domain, but a Software Testing Ontology in UML for a Software Growth Environment of Web-Based Applications was proposed in [Huo et al., 2003]. We have included these terms in relation with the concept Test of *V*&*V* techniques in the *VVO*.

5 Conclusion and future work

This paper aims to propose a general conceptualization of the *V*&*V* Ontology. The *VVO* includes foundations for formalisms and *V*&*V* techniques, but it is designed actually for knowledge sharing purposes.

This ontology can be developed farther in several directions. In the near future, the System concept hierarchy should be developed with various system types such as Real-time and Concurrent systems, in addition to the corresponding links with formalisms (relations such as Concurrent System can be *described* with the Pi-Claculus formalism and a Real-time system can be described with Automata formalism). The accepted operations for each type of system must be also included such as: the Automata formalism *supports* Parallel and Hierarchical composition of states.

The resulting *VVO* will be validated by linking it with more existing *V*&*V* tools and experimenting with various systems. The planed work is too much for few researchers. For the *VVO* to evolve, it is required that the *V*&*V* community works in a more synergistic way, explicitly building on, populating and extending this version of *VVO*.

Acknowledgements

This work was funded by the European Union and the French DGCIS through the ARTEMIS Joint Undertaking inside the CESAR project. A preliminary version of this work was presented at the KEOD'2010 conference. Thanks to Andres TOOM for reading and correcting this paper.

[7] http:// www.biomath.jussieu.fr/Menelas/Ontologie/html/

References

[Alur et al., 1993] Alur, R., Courcoubetis, C., Henzinger, T., and Ho, P. (1993). Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. *Hybrid systems*, pages 209–229.

[Bengtsson and Yi, 2003] Bengtsson, J. and Yi, W. (2003). Timed automata: Semantics, algorithms and tools. *Lectures on Concurrency and Petri Nets*, pages 87–124.

[Bodenreider and Burgun, 2005] Bodenreider, O. and Burgun, A. (2005). Biomedical ontologies. *Medical Informatics*, pages 211–236.

[Büchi, 1960] Büchi, R. J. (1960). Weak Second-Order Arithmetic and Finite Automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6(1-6):66–92.

[Castells et al., 2004] Castells, P., Foncillas, B., Lara, R., Rico, M., and Alonso, J. (2004). Semantic web technologies for economic and financial information management. *The Semantic Web: Research and Applications*, pages 473–487.

[Cervelle et al., 2010] Cervelle, J., Formenti, E., and Guillon, P. (2010). Ultimate Traces of Cellular Automata. *Arxiv preprint arXiv:1001.0251*.

[Chaki et al., 2004] Chaki, S., Clarke, E., Ouaknine, J., Sharygina, N., and Sinha, N. (2004). State/event-based software model checking. In *Integrated Formal Methods*, pages 128–147. Springer.

[D'Argenio and Katoen, 2005] D'Argenio, P. and Katoen, J. (2005). A theory of stochastic systems part I: Stochastic automata. *Information and computation*, 203(1):1–38.

[Fonou-Dombeu and Huisman, 2011] Fonou-Dombeu, J. and Huisman, M. (2011). Combining ontology development methodologies and semantic web platforms for e-government domain ontology development. *International Journal of Web & Semantic Technology (IJWesT)*, 2(2):12–25.

[Gomez-Perez et al., 1991] Gomez-Perez, A., Fernández-López, M., and Corcho, O. (1991). Ontological engineering. *AI Magazine*, 36:56.

[Hilliard, 2000] Hilliard, R. (2000). Ieee-std-1471-2000 recommended practice for architectural description of software-intensive systems. *IEEE, http://standards. ieee. org*.

[Huo et al., 2003] Huo, Q., Zhu, H., and Greenwood, S. (2003). A multi-agent software environment for testing Web-based applications. *COMPSAC-NEW YORK-*, pages 210–215.

[Lawson, 2005] Lawson, M. (2005). Finite automata. *Handbook of networked and embedded control systems*, pages 117–143.

[McGuinness et al., 2004] McGuinness, D., Van Harmelen, F., et al. (2004). OWL web ontology language overview. *W3C recommendation*, 10:2004–03.

[Mikk et al., 1997] Mikk, E., Lakhnechi, Y., and Siegel, M. (1997). Hierarchical automata as model for statecharts. *Advances in Computing Science—ASIAN'97*, pages 181–196.

[Motik et al., 2009] Motik, B., Patel-Schneider, P., Parsia, B., Bock, C., Fokoue, A., Haase, P., Hoekstra, R., Horrocks, I., Ruttenberg, A., Sattler, U., et al. (2009). Owl 2 web ontology language: Structural specification and functional-style syntax. *W3C Recommendation*, 27.

[Perrin, 2004] Perrin, D. (2004). *Infinite words: automata, semigroups, logic and games*. Academic Press.

[Visser and Bench-Capon, 1998] Visser, P. and Bench-Capon, T. (1998). A comparison of four ontologies for the design of legal knowledge systems. *Artificial Intelligence and Law*, 6(1):27–57.